МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
ТЕМА: Кодирование: статическое Хаффмана

Студент гр. 8304 Мухин А. М.

Преподаватель Фирсов М. А.

Санкт-Петербург
2019

Цель работы.

Реализация и экспериментальное машинное исследование алгоритмов кодирования (Фано-Шеннона, Хаффмана), быстрого поиска на основе БДП или хеш-таблиц, сортировок.

Задание.

Кодирование: статическое Хаффмана. 3 вариант.

Выполнение работы.

Работа программы начинается со считывания строки, которую необходимо закодировать. Далее создаётся ассоциативный массив, который посредством функции std::map<std::wstring, int> get_list_count_letter(std::wstring) заполняется ключами из элементов строки и значениями по этим ключам, являющимися количеством вхождений данного ключа во входную строку.

Далее, для более удобного оперирования с этими данными, значения перегоняются в вектор типа std::vector<std::pair<std::wstring, int>> для дальнейшей удобной сортировки и изменения значений элементов этого вектора.

Сортировка происходит с помощью функции стандартной библиотеки std::sort(), которая принимает итератор на начало веткора, на его конец и функцию компаратора. В нашем случае функцию компаратора заменяет лямбда функция.

Далее идёт главный цикл программы, который закончится, когда в нашем векторе останется только один элемент(все символы последовательности, как одна строка). Также во время одной итерации определяется итератор на последний и предпоследний элемент, которые между собой суммируются и возникает новый элемент, который и вставляется в наш вектор, а два последних удаляются. Также, именно тут строится код каждого элемента, если частота последнего элемента равна частоте предпоследнего, то в начало кода для последнего дописывается 1, а предпоследнего 0. Если не равны, ноборот.

Также следует обратить внимание на функцию поиска места, куда необходимо вставить новый элемент. std::vector<std::pair<std::wstring, int>>::iterator decision(std::vector<std::pair<std::wstring, int>>&, int) – возвращает итератор на найденную позицию, перед которой необходимо произвести вставку. Реализована довольно просто и не требует подробных объяснений.

Наконец, после того, как все манипуляции с нашими данными выполнены, остаётся только записать данные в два выходных файла. В файле symbol_code.txt будет находится информация в виде <символ>:<его код>, а в файле coding_message.txt будет находится закодированное сообщение, которое подавалось на вход.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№	Входные данные	Выходные данные
1.	вафвфывавыфывафыва фывафывафывафвыаф	11 00 10 11 10 01 11 00 11 01 10 01 11 00 10 01 11 00 10 01 11 00 10 01 11 00 10 01 11 00 10 11 01 00 10
2.	втыыАИПкиаипгьугцм кЦМКУЦицукиецукецк	10111 00010 0101 0101 10000 10001 10100 110 011 10110 011 00001 0011 00011 1110 0011 1111 00000 110 0010 10011 10010 10101 0010 011 1111 1110 110 011 0100 1111 1110 110 0100 1111 110
3.	ропылвапSDFsfgары35 635ПЫИАпdfgshSsfdghsi	11100 00011 1011 11101 00010 00001 0111 1011 0100 10101 10000 1111 1100 1101 0111 11100 11101 0010 0011 10100 0010 0011 10011 00000 10010 1011 0101 1100 1101 1111 0110 0100 1111 1100 0101 1101 0110 1111 10001
4.	qwertyPBAП452DFGва ывапываы3789рлыврао	00000 00011 10111 00001 00010 00100 01000 00110 00101 00111 11001 10000 11010 10100 10101 10110 1110 011 1111 1110 011 01011 1111 1110 011 1111 11000 10001 10010 10011 11011 01001 1111 1110 11011 011 01010
5.	24763527834652783465 78623273465823475234	00 110 101 011 111 100 00 101 010 111 110 011 100 00 101 010 111 110 011 100 101 010 011 00 111 00 101 111 110 011 100 010 00 111 110 101 100 00 111 110

Выводы.

В ходе данной лабораторной работы мы научились статически кодировать текстовое сообщение методом Хаффмана. Изучили как ведут себя широкие символы, потоки для них, а также контейнеры для их хранения и обработки. Поработали с итераторами и контейнера стандартной библиотекой шаблонов, такими как unordered_map, map, vector, string.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Static_Haffman.cpp

```
#include <set>
#include <iostream>
#include <map>
#include <vector>
#include <algorithm>
#include <locale>
#include <string>
#include <windows.h>
#include <unordered_map>
#include <fstream>

std::map<std::wstring, int> get_list_count_letter(std::wstring local_string) {
	std::map<std::wstring, int> set;
	for (auto c : local_string) {
		std::wstring tmp_string = { '\0' };
		tmp_string = c;
		if (set[tmp_string] == 0) {
			set[tmp_string] = 1;
		} else {
			set[tmp_string]++;
		}
	}
	return set;
}

std::vector<std::pair<std::wstring, int>>::iterator decision(std::vector<std::pair<std::wstring, int>>& arr, int b) {
	for (std::vector<std::pair<std::wstring, int>>::iterator it = arr.begin(); it != arr.end(); it++) {
		if ((*it).second <= b) {
			return it;
		}
	}
	return arr.end();
}

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	std::wstring input_string = {};
	std::vector<std::pair<std::wstring, int>> arr;

	std::getline(std::wcin, input_string);
	std::map<std::wstring, int> set = get_list_count_letter(input_string);

	for (auto iter : set) {
	    arr.emplace_back(std::pair<std::wstring, int>(iter.first, iter.second));
	}
```

```
          std::sort(arr.begin(),  arr.end(),  [](std::pair<std::wstring,
int> a, std::pair<std::wstring, int> b) {
                                                                  return
a.second > b.second;
                                                              }
          );

          int sum_frequency = 0;
          std::wstring sum_str = {};

          if (arr.size() > 2) {
               std::unordered_map<wchar_t, std::wstring> result = {};
               while (arr.size() != 1) {
                    // определяем последний и предпоследний элементы
                    auto last = --arr.end();
                    auto penultimate = arr.end() -= 2;

                    //  считаем  их  общую  частоту  и  определяем  новый
элемент
                    sum_frequency       =       (*last).second       +
(*penultimate).second;
                    sum_str = (*last).first + (*penultimate).first;

               // строим путь до элементов
                    // проверка на равенство: тогда должны поменяться
местами последний и предпоследний элементы, но так как
                   // далее проводятся операции только через итераторы,
мы просто поменяет местами эти итераторы)))
                    if ((*last).second == (*penultimate).second) {
                         auto tmp = std::move(last);
                         last = std::move(penultimate);
                         penultimate = std::move(tmp);
                    }
                    // операции с последним элементом
                    if ((*last).first.size() == 1) {
                         result[(*last).first[0]] = L"0";
                    } else {
                         for (auto c : (*last).first) {
                              result[c] = L"0" + result[c];
                         }
                    }

                    //операции с предпоследним элементом
                    if ((*penultimate).first.size() == 1) {
                         result[(*penultimate).first[0]] = L"1";
                    } else {
                         for (auto c : (*penultimate).first) {
                              result[c] = L"1" + result[c];
                         }
                    }

                    //  вставляем  новый  элемент  с  общей  частотой  в
нужное место и удаляем два последних
                    arr.emplace(decision(arr,           sum_frequency),
std::pair<std::wstring, int>(sum_str, sum_frequency));
                    arr.erase(arr.end() -= 2, arr.end());
               }
```

```
                std::wofstream symbol_code("symbol_code.txt");
                std::wofstream coding_message("coding_message.txt");

                // вывод значений символ : его код
                for (auto pair : set) {
                     symbol_code    <<    pair.first    <<    ":"    <<
result[pair.first[0]] << std::endl;
                }

                // вывод закодированного сообщения
                for (auto symbol : input_string) {
                     coding_message << result[symbol] << L" ";
                }
           }
           return 0;
     }
```